Prof. Dr. Manfred Schmidt-Schauß
Institut für Informatik

Fachbereich Informatik und Mathematik/ Institut für Informatik
Johann Wolfgang Goethe-Universität Frankfurt am Main

Einführung in die funktionale Programmierung

Wintersemester 2019/2020

Aufgabenblatt Nr. 1

Abgabe: Montag 21. Oktober 2019 vor der Vorlesung

Bitte unbedingt Wunsch-Übungsgruppe auf Abgabe notieren!

Aufgabe 1 (15 Punkte)

Sei s der folgende Ausdruck des Lambda-Kalküls:

$$s := (\lambda x, y, z.x\ y\ z)\ \lambda x, y.x\ \lambda y, z.z$$

a) Schreiben Sie s ohne abkürzende Schreibweisen und mit voller Klammerung auf.(5 Punkte)

b) Geben Sie s in Haskell-Notation an und geben Sie den entstandenen Ausdruck im Haskell-Interpreter `ghci` ein. (5 Punkte)

c) Geben Sie einen α-äquivalenten Ausdruck zu s an, der die *Distinct Variable Convention* erfüllt. (5 Punkte)

Aufgabe 2 (20 Punkte)

Die Collatz-Funktion kann in Haskell programmiert werden durch:

```
collatz 1 = 1
collatz n = if even n
             then collatz (n `div` 2)
             else collatz (3*n+1)
```

Fügen Sie obigen Programmcode in eine Datei namens `Collatz.hs` ein und laden Sie das Programm anschließend im `ghci`[1]. Testen Sie anschließend die Funktion für mindestens 10 verschiedene Werte.

Zum Debuggen würde man gerne die Zwischenwerte von n bei jedem Rekursionsschritt sehen. Die Haskell-Bibliothek `Debug.Trace` stellt die Funktion `trace :: String -> a -> a` bereit, die zwei Argumente erwartet und das erste (als Debug-Output) ausdruckt und anschließend das zweite als Resultat liefert.

Verwenden Sie `trace` in geeigneter Weise, um die Zwischenwerte von n auf der Konsole auszudrucken. Testen Sie die modifizierte Funktion erneut.

[1] Wenn Sie nicht wissen, wie das funktioniert, verwenden Sie die Hilfe des `ghci`, indem Sie `:?` im `ghci` eintippen

Hinweise:

- Das Modul `Debug.Trace` kann eingebunden werden, indem Sie am Anfang des Quelltexts die Zeile

  ```
  import Debug.Trace
  ```

  einfügen.

- Zum Konvertieren einer Zahl in einen `String` können Sie die vordefinierte Funktion `show` verwenden.

Aufgabe 3 (15 Punkte)

Lambda-Abstraktionen sind *anonyme Funktionen*, also Funktionen die keinen Namen haben. Z.B. könnten wir für das Haskell-Programm

```
fun1 x y   = x + 2*y
fun2 x y z = (fun1 x z) - (fun1 (2*x) y)
fun3 f g x = f (g x) (g x)
```

und den Aufruf `fun1 (3-1) 5` auch den gleichwertigen Ausdruck

```
(\x y -> x + 2*y) (3-1) 5
```

eingeben.

a) Geben Sie für die folgenden Aufrufe gleichwertige Ausdrücke an, ohne die Funktionen `fun1`, `fun2` und `fun3` zu verwenden und geben Sie die entstandenen Ausdrücke im `ghci` ein:

   i) `fun1 (2*6) (3*8)` (3 Punkte)

   ii) `fun3 fun1 (fun2 1 2) 3` (5 Punkte)

   iii) `fun1 (fun2 1 2 3) (fun1 2 7)` (5 Punkte)

b) Welche Probleme entstehen, wenn Sie `collatz 5` (siehe Aufgabe 2) in einen solchen Ausdruck überführen möchten? (2 Punkte)